Supporting PDF accessibility evaluation: early results from the FixRep project

Andrew Hewson[1] and Emma Tonkin[2]

[1]a.hewson@ukoln.ac.uk, UKOLN, University of Bath, UK
[2]e.tonkin@ukoln.ac.uk, UKOLN, University of Bath, UK

Abstract: The aim of this paper is to present results from a pilot study exploring automated formal metadata extraction in accessibility evaluation. Information about some types of accessibility may make up part of the formal metadata for a document. As the importance of document accessibility has become more widely accepted and relevant legislation has been identified and characterised, the possibility of storing information about document accessibility as part of the formal metadata held by the system has become more attractive. This is useful in order to provide a starting-point for an accessibility assessment. This study reviews accessibility issues linked to the PDF format in use. We demonstrate a prototype created during the FixRep project, that aims to support capture, storage and reuse of accessibility information where available, and to approach the problem of reconstructing required data from available sources. Finally, we discuss practical use cases for a service based around this prototype.

Keywords: automated metadata extraction; accessibility; text analysis

1. Introduction

The aim of this paper is to present results from a pilot study run within the FixRep project, which aims to examine and enhance existing techniques and implementations for automated formal metadata extraction. Formal metadata, such as filetype, title, author and image captions (by comparison to subject metadata, which usually draws on information extrinsic to the document itself) is mostly intrinsic to the document and its citation. Some formal metadata is collected by almost all repositories. Information about some types of accessibility may make up part of the formal metadata for a document.

In this study, we began with exploration of the PDF format, widely used across a large number of contexts of use in the digital library environment. Web-based uses of relevance to digital libraries for example include: forms; printable versions of resources, particularly those (such as PowerPoint documents) for which there is no free viewer available; and pre-prints of papers and articles. It is not always widely recognised that two different encodings for a given PDF may have entirely different properties as regards accessibility. A well-formed document with extensive annotation may be quite usable via a screen reader. Another may be entirely unreadable with accessibility software. When printed or viewed on screen, the two may appear identical.

A variety of software packages and services exist that aim to support the accessibility assessment of PDF documents. In general, what is meant by 'accessible' PDF files is 'tagged', or 'structured' PDFs. These are a structured, textual representation of the PDF, which are intended for use by screen readers. These represent additional information, so the creation of a tagged PDF usually requires additional work. It is often simpler from an accessibility viewpoint to represent documents as HTML (Clark, 2005). However, where PDFs exist, it is possible to assess just how usable or accessible those documents are. We introduce a prototype written during the FixRep project, that aims to support

the capture, storage and reuse of accessibility information where it is available, and to reconstruct required data from available sources where it is not. Finally, we discuss possible use cases for this prototype in a practical repository context, exploring how and where automated evaluation methods such as these can be usefully applied.

Document accessibility in self-deposit repositories

In repositories that are centrally managed, it is often possible to put reasonably strict requirements in place, and enforce them with a reasonable degree of success. However, this is rapidly complicated by widening the eligibility and encouraging a greater degree of self-deposit activity; in effect, a greater breadth of document types and content implies a wider variety in the resulting document set. The well-ordered, carefully managed repository lies at one extreme; at the opposite extreme is a chaotically organised file-store. In most cases, the reality lies somewhere between these extremes.

As a result of these practical limitations, it is perhaps inevitable that details such as complete and appropriate representative metadata or the use of appropriate mechanisms to ensure that accessibility requirements are met should be approached opportunistically – that is, 'nice to have if they're available'. But there are considerable legal and practical considerations that should represent an encouragement to users and repository managers alike to look upon accessibility as a concern, as well as a realistically achievable goal.

The legal aspects of accessibility are well-known and documented, at least from a UK perspective. Bailin (2007) notes that in 2002, the European Parliament "*set the minimum level of accessibility for all public sector websites3 at Level Double-A. However, a… survey of public sector services showed that 70% of websites in the European Union failed to conform to Level-A of the W3C guidelines.*" As the importance of document accessibility has become more widely accepted and relevant legislation, such as the Disability Discrimination Act (1995) in the UK, has been identified, the possibility of storing information about document accessibility as part of the formal metadata held by the system has become more attractive. This is useful for various purposes, primarily in order to provide a starting-point for an accessibility assessment, leading into a triage process.

The practical considerations mentioned are to do with the availability of the document for reuse. A badly formed or non-machine-readable document placed online is of marginal practical use. Obviously, it is better to place it online than to fail to publish it at all. However, if there were a review mechanism enabling users to be aware of the usability issues, then they would be in a better position to review their documents at an early stage, and to decide for themselves whether they prefer to accept the limitations of the current expression of the document, or to recreate an alternative or additional document to place onto the repository, to replace or supplement the original file. What is suggested here is not strict validation, but support for user-level review and triage.

What's in a repository?

Our research questions are the following: at present, what span of content appears in a document repository that enables user deposit? Does this variation in document format imply a reduction in accessibility, what sort of reduction, to whom, and to what extent? Is it possible for us to automatically identify issues that may be of particular concern, or for us to identify good practice where it is used?

As is often the case, it is important to separate that which is simply non-optimal from 'show-stopper' issues. The former are of some concern, most specifically in terms of potential impact on preservation and longer-term accessibility, whilst the latter may be of immediate concern or, at least, pose a significant enough difficulty to request the user to review the issue as a matter of some urgency. An unreadable or corrupt document, for example, is a 'show-stopper'. A PDF that is missing fonts or has certain issues that impair formatting or reduce readability is problematic, but it is likely to be possible to work with it for at least some users. A PDF that is simply a collection of images is relatively unproblematic for sighted users, but poses significant difficulties for the non-sighted.

In this paper, we characterise the problems that are detectable using our prototype software, and compare this approach to a more formal mechanism of accessibility-checking. We characterise the papers stored within one institutional repository, and discuss the potential impact of institutional repository policies such as the placement of a cover page onto the head of each PDF.

2. Methodology

As part of the FixRep project, a prototype has been developed for analysis of PDFs. This extracts information about the document in a number of ways:

- **Header and formatting analysis**: information about the PDF can be extracted from the document headers, such as:
    - The version of the PDF standard in use
    - Whether certain features, such as PDF tagging, are declared to be in use
    - The software used to create the PDF
    - The publisher of the PDF
    - The date of creation and last modification
- **Information from the body of the document**: information about the content of the document, such as:
    - Whether images or text could be successfully extracted from the document and, if they could, information about those data objects.
    - If any text could be extracted from the object, further information such as the language in which it appeared to be written and the number of words in the text
- **Information from the originating filesystem**: metadata from the originating filesystem such as document path, size, creation date, etc.

This, then, is a much simplified form of metadata extraction that places little emphasis on complex content analysis, but more emphasis on the different object types stored within the document and the format of the document.

The prototype has been developed in Perl using a number of well-known tools: *pdfinfo*, *pdftotext,* and *pdfimages*. It also uses a number of CPAN modules in order to identify language, tokenise, and return relevant metadata about images. The service API is designed along the lines of a REST service, which is to say a simple HTTP-based service that makes use of simple, standard web protocols to surface relevant functionality. It makes use of syntax calls such as the following:

Document submission:
http://fixrep.ukoln.ac.uk/pdfAssay/=/link/http://example.com/a.pdf
Retrieval of a single component content:
http://fixrep.ukoln.ac.uk/pdfAssay/=/retrieve/unique-id-of-component

This prototype has been written primarily for the purpose of supporting rapid development of applications depending on access to components or content of PDF files, such as graphics, content, and format metadata. It is a sister service to formal metadata extraction systems.

We chose to explore the OPUS repository, managed by the University of Bath, UK. In order to enable this, we began by spidering the site in order to identify all the PDFs available on the site. These were then cached offline, and, via a batch processing job, were passed to the service prototype for analysis. The responses were added into a mySQL database in order to enable the results to be analysed. The data analysis process was, as this is the first pilot study, completed largely by hand – that is, through a handcrafted series of SQL queries. We envisage that in future it will be possible to largely automate this process.

3. Results

A proportion of the documents (approximately 20%) were not successfully processed during the first sweep for a variety of reasons. The rest of the statistics given here relate to those files that completed at least partial processing (Fig. 1).

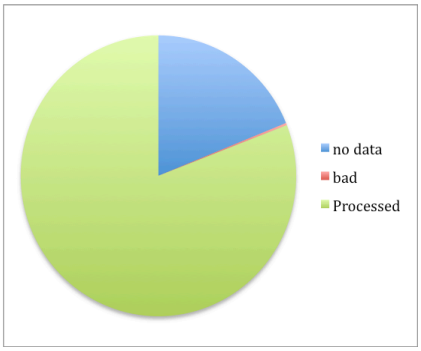

Figure 1: PDF harvested from Opus

More detailed statistics are given in the following figure (Fig. 2) which pulls out each category of metadata collected and reviews the proportion of the documents for which the terms could be extracted.

Some of these terms are extracted directly via the software packages previously mentioned, such as 'Creator', 'PDF-version', and 'Author'. Others are generated by the software prototype, such as the guessed language and the number of words in the document. It is important to realise that in the context of PDF metadata, terms such as 'Creator' do not have the meaning that would be ascribed to them in the Dublin Core standard, for example – or if they do, it is a coincidence. In PDF, for example, the 'Author' keyterm exists to describe the individual who authored the document; 'Creator' refers to the software used to create the content (the editor, such as Microsoft Word), whilst 'Producer' is the software used to generate the PDF – such as a printer driver or a PDF creation/format transformation program.

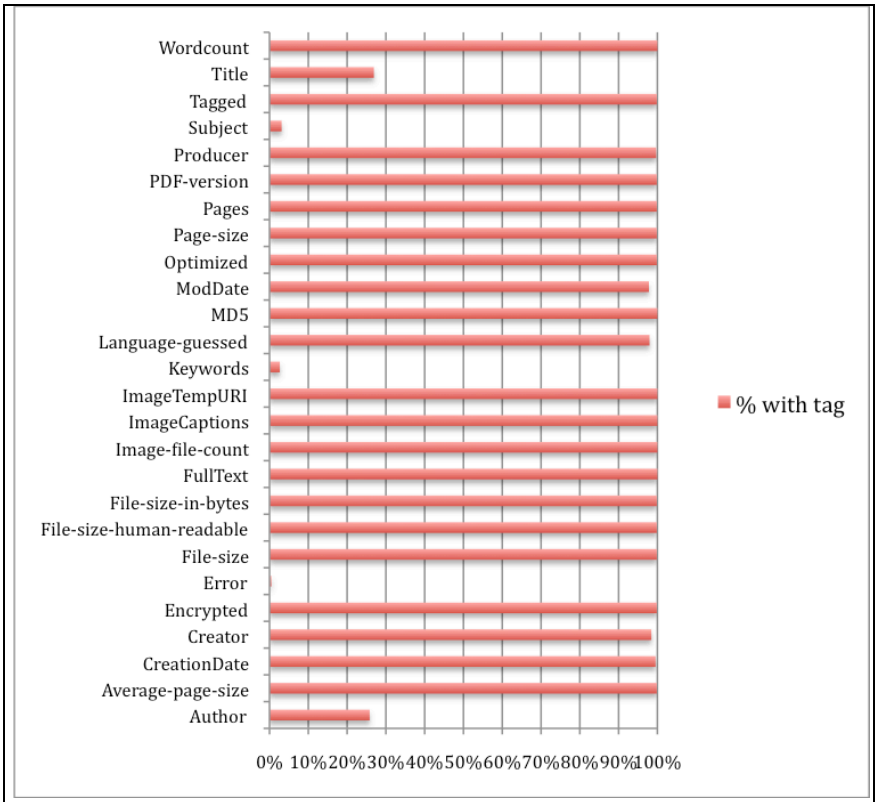

Figure 2: Percentage of PDF with Tag

It is important to note that the 'traditional' metadata (author, title and keywords, for example) are sparsely populated in this dataset. This speaks for the importance of an external metadata record containing this information.

While 'traditional' values might be missing, overall the average number of metadata tags utilised and the statistical mode of their use correspond closely (at approximately 21 per document), so that we can infer somewhat consistent usage of metadata tags within PDFs; this is not surprising as many of these are defined as required within the PDF standard. A few of these are also generated by our software to enhance our understanding of the content, such as 'Language-guessed', and these will be present in the vast majority of cases.

Fig. 3 & Fig. 4 below, show the distribution of PDF versions in use, and for those where the 'Tagged' metadata was provided (the vast majority), the proportion of (structured) PDFs was 9.35%. This means that only 10% of all PDFs processed have any likelihood of conforming to accessibility guidelines, and even then we would require further content level analysis to evaluate the extent to which they do indeed conform.

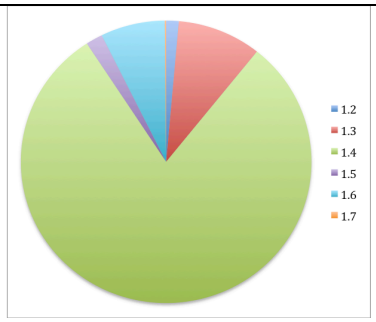

Figure 3: PDF Versions

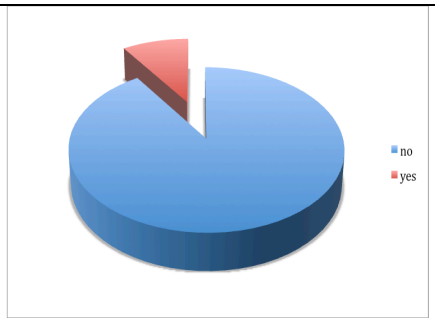

Figure 4: Structured PDF Proportion

Fig. 5 shows the distribution of 'Producer' applications. These are essentially alternative format-to-PDF conversion applications. This statistic offers us our first clear hint that something is influencing the distribution of conversion applications; as can be seen in the pie chart superimposed, there is one utility used by around two-thirds of file creation processes! As with Fig. 6, it has not been represented in the main bar chart as it is disproportionately large and damages visibility of the main distribution. This does not appear to fit the distribution that we would expect; inspection of these files shows that each one has been recreated with a prepended cover sheet. It seems that the producer in these instances has been 'reset' or overwritten by this prepending process. The same is true of the PDF creation tools (see Fig. 6), which show a similar distribution, although with a different application. It appears that the 'pdftk' tool is used to concatenate a cover-sheet that was itself generated using the 'itext-paulo' library.

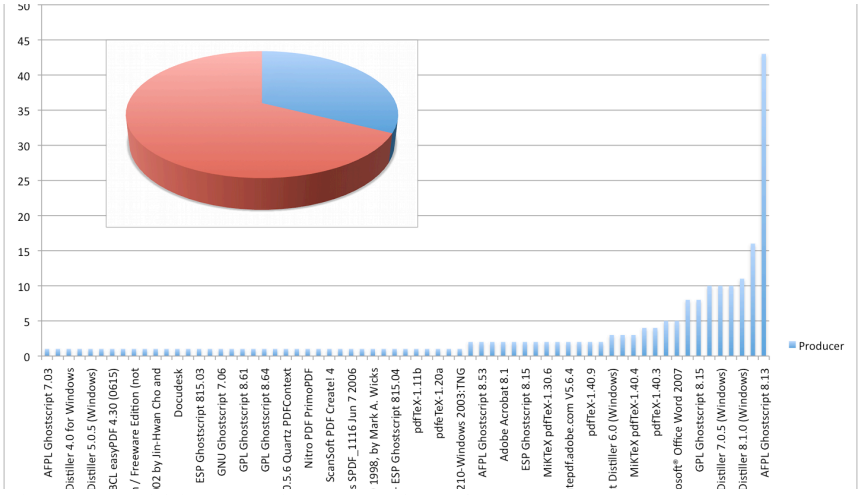

Figure 5: Utilities used in production of PDFs. Insert chart demonstrates relative popularity of utilities named in graph, and most popular utility (itext-paulo)

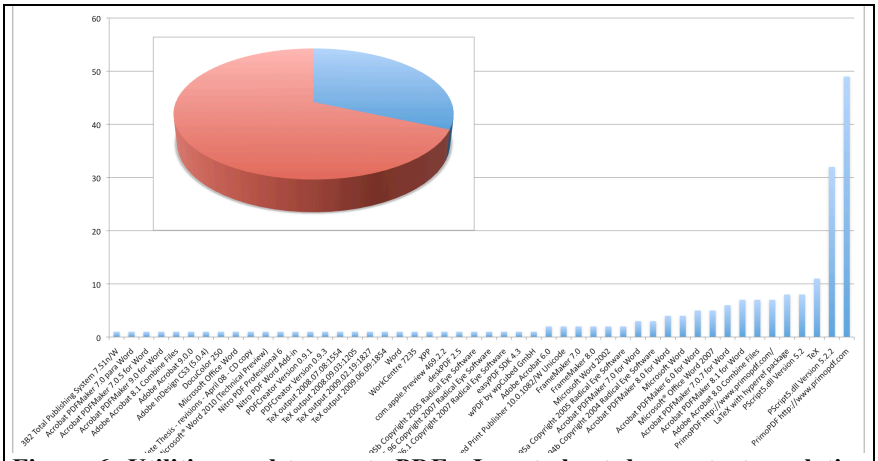

Figure 6: Utilities used to create PDFs. Insert chart demonstrates relative popularity of PDF creator utilities named in graph, versus the most popular utility (pdftk 1.12)

PDF files that generated errors

The following table shows the types of errors encountered in working with this content and the number of files involved, and whether they are recoverable for use in the context of the project.

Error type	Count	Recoverable
Copying intentionally disabled	13	No
Structural errors relating to fonts	5	Yes
Damaged / corrupt file	1	No
Structural damage	2	Yes

The damaged files in total only represent about 3% of all files that could be processed. The surprising outcome of this is that the largest single cause of rendering files unusable for the purposes of machine processing is the use of copy protection to limit content extraction and reuse.

Exploring the full text extracted from each document by means of inspecting the first lines of each document demonstrates another of these unexpected distributions seen earlier (i.e. the 'Creator' and 'Producer') and again this is related to the addition of cover sheets to the PDFs: inspection shows that two-thirds of documents ostensibly begin with the title "*University of Bath Opus Online Publications Store*". A human reader will recognise this page as a cover sheet, and thus skip forward to the main content of the document. However it is arguable that the same cannot be said of automated processes without prior knowledge of this phenomenon.

The impact of cover pages on document indexing services

Many repositories, including but by no means limited to the University of Bath repository, have developed or identified a means of adding a cover sheet to each document within the repository. This has potential for positive impact, for example, as a means of clearly indicating the provenance of an item (Puplett, 2008). As can be seen in Fig. 7, Google Scholar does not necessarily recognise the cover sheet for what it is, and this has negative implications for effective indexing and retrieval.

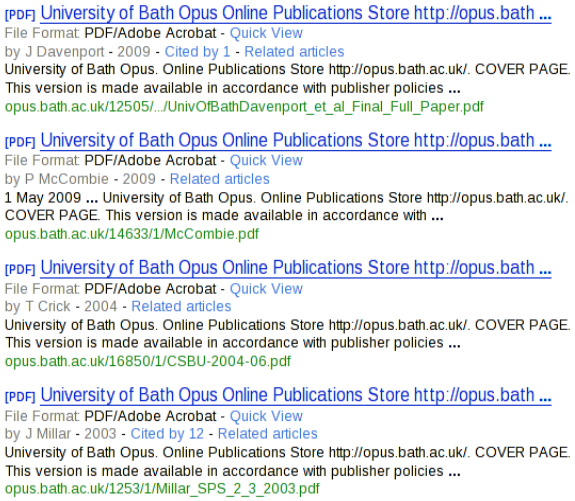

Figure 7: Indexing on documents with cover pages

4. Conclusions

We find that 10% of documents implement tagging; this indicates that there may well be a number of authors who are potentially able to develop well-structured PDFs. This is a higher proportion than was expected and is certainly a cause for optimism for human accessibility. However, the addition of a cover sheet has caused a number of issues beyond those that are usually encountered with the PDF format (ie. font problems, file corruption, etc). This limits the ability for automated processes to make use of this information, and could therefore be said on the level of automated indexing and other software access (such as conversion) to be a retrograde step. If this becomes common practice it may be necessary to review both the assumptions under which automated systems are developed, and perhaps the rationale that lead us to make use of cover sheets in this context.

References

Adobe (2010). Acrobat built-in accessibility checker. Retrieved April 20, 2010, from http://www.adobe.com/accessibility/products/acrobat/faq.html

Bailin, A (2007). Delivering inclusive websites: user-centred accessibility. Retrieved April 20, 2010, from http://www.cabinetoffice.gov.uk/media/cabinetoffice/corp/assets/publications/government_it/consultations/pdf/delivering_inclusive_websites1.pdf

Clark, J. (2005). Facts and Opinions about PDF Accessibility. A List Apart. ISSN:1534-0295

Coonin, B. (2002), "Establishing accessibility for e-journals: a suggested approach", Library Hi Tech, Vol. 20 No.2, pp.207-13.

Poppler (2010). A PDF rendering library. http://poppler.freedesktop.org/

Puplett, D. (2008). Version Identification: A Growing Problem. *Ariadne* 1(54), January 2008. ISSN 1361-3200

Richardson, L.; Ruby, S. (2007), *RESTful Web Services*, O'Reilly (published (May 8, 2007)), ISBN 0596529260

Web Content Accessibility Guidelines (WCAG) 2.0 - W3C Recommendation 11 December 2008. Retrieved April 20, 2010 from http://www.w3.org/TR/2008/REC-WCAG20-20081211/